CLAIMS

1. (Currently amended) A method for allocating core processor bandwidth, comprising:

detecting an interrupt service request;

storing into an instruction cache interrupt service instructions in response to detecting the interrupt service request;

forming an instruction stream sequence within the instruction cache, the instruction stream sequence including mainline program instructions and the interrupt service instructions resulting in allocating core processor bandwidth between the interrupt servicing and mainline program instructions in accordance with capacity of the core processor;

fetching instructions from the instruction stream sequence within the instruction cache;

~~inserting interrupt servicing instructions into an instruction queue mechanism in response to detecting the interrupt service request, wherein the instruction queue mechanism includes an instruction cache and an instruction fetch unit;~~

~~inserting the interrupt servicing instructions into mainline program instructions within the instruction queue mechanism resulting in allocating core processor bandwidth between the interrupt servicing and mainline program instructions; and~~

decoding ~~the mainline program and the interrupt servicing~~ instructions from the instruction stream sequence into micro-ops;

executing the micro-ops in one or more out-of-order execution units;

recycling the executed micro-ops for optional re-execution in the one or more out-of-order execution units by retiring the executed micro-ops including those micro-ops representing the interrupt servicing instructions to the instruction cache.

2. (Canceled)

3. (Canceled)

4. (Previously presented) The method of claim 1 comprising retiring the executed micro-ops to the instruction cache in order.

5. (Currently amended) The method of claim 1 where detecting comprises detecting plural interrupts by prioritizing the plural interrupts and ~~inserting~~ storing one or more instances of the interrupt servicing instructions into the instruction ~~queue mechanism~~ stream sequence within the instruction cache in accordance with one or more predefined interrupt servicing allocation criteria.

6. (Previously presented) The method of claim 5 where the one or more allocation criteria include the priority of the interrupts and the capacity of the processor to allocate bandwidth to interrupt servicing.

7. (Previously presented) The method of claim 6 where the prioritizing is dynamically responsive to changing allocation criteria.

8. (Previously presented) The method of claim 1 where the processor bandwidth is allocated to interrupt servicing without flushing the instruction cache.

9. (Currently amended) The method of claim 1 where the detecting is performed by an interrupt processor, which after the detecting, the method comprising:

at the interrupt processor, determining whether the detected interrupt service request is of a priority meeting one or more defined high-priority criteria and if so then signaling the core processor to perform the ~~inserting~~ storing; and

at the core processor, responding to said signaling by performing said ~~inserting~~ storing and ~~said~~ processing.

10. (Currently amended) A method comprising:

detecting an interrupt service request;

storing into an instruction cache interrupt service instructions in response to detecting the interrupt service request;

forming an instruction stream sequence in the instruction cache, the instruction stream sequence including mainline program instructions and the interrupt service instructions resulting in allocating core processor bandwidth between the interrupt servicing and mainline program instructions in accordance with capacity of the core processor;

fetching instructions from the instruction stream sequence within the instruction cache;

~~inserting interrupt servicing instructions into an instruction queue mechanism in response to detecting the interrupt service request where inserting the interrupt servicing instructions into the instruction queue mechanism results in a core processor concurrently in-line staging the interrupt servicing instructions and other program instructions;~~

executing the interrupt servicing instructions and the other program instructions from the instruction stream sequence within the instruction ~~queue mechanism~~ cache;

determining that a natural processor context switch is imminent, said determining including analyzing one or more instructions within the instruction ~~queue mechanism~~ cache for context switch-inducing instructions;

signaling ~~the~~ an interrupt processor to make ready the highest priority interrupt service request; and

signaling ~~the~~ an instruction ~~queue mechanism~~ fetch unit to fetch the readied interrupt service request in advance of a context switch responsive to the determining;

where the detecting is performed by ~~an~~ the interrupt processor.

11. (Currently amended) A processor, comprising:

an instruction cache to store an instruction stream sequence, the instruction stream sequence including mainline program instructions and interrupt service instructions resulting in allocating processor bandwidth between the interrupt servicing and mainline program instructions in accordance with capacity of the processor;

an interrupt handler to detect the interrupt service instructions;

a fetch unit to fetch instructions from the instruction stream sequence ~~mainline program and interrupt servicing instructions from~~ within the instruction cache;

a decode unit to decode the instruction stream sequence including mainline program and interrupt servicing instructions into micro-ops;

a dispatch unit to schedule the micro-ops for execution;

an execute unit to execute the micro-ops;

a retirement unit to retire executed micro-ops including the micro-ops corresponding to the interrupt instructions back to the instruction cache; and

where the interrupt handler signals the fetch and decode units to insert micro-ops corresponding to the interrupt servicing instructions into ~~an~~ the instruction stream sequence within the instruction cache without flushing the instruction cache.

12. (Currently amended) The processor of claim 11

where the processor supports plural hardware interrupt inputs;

where the interrupt-handler includes an interrupt concentrator to determine priority among the plural hardware interrupt inputs and to schedule the plural hardware interrupt inputs; and

where the interrupt-handler is adapted to instruct the fetch and decode units such that plural instances of such decoded micro-ops are inserted, each instance representing one or more corresponding sets of the interrupt servicing instructions, into ~~a normal~~ the instruction stream sequence for serial scheduling and execution of the plural instances of such decoded micro-ops by the dispatch and execute units in accordance with a determined priority of the plural hardware interrupt inputs.

13. (Currently amended) A processor comprising:

an instruction cache to store an instruction stream sequence, the instruction stream sequence including mainline program instructions and interrupt service instructions resulting in allocating processor bandwidth between the interrupt servicing and mainline program instructions in accordance with capacity of the processor;

a fetch unit to fetch instructions from the instruction stream sequence ~~mainline program and interrupt servicing instructions from~~ within the instruction cache;

an interrupt handler to detect the interrupt servicing instructions;

a decode unit to decode instructions from the instruction stream sequence including mainline program and interrupt servicing instructions into micro-ops;

a dispatch unit to schedule the micro-ops for execution;

an execute unit to execute the micro-ops;

a context switch prediction unit coupled with the fetch and decode units to predict a naturally occurring context switch and to signal the interrupt handler upon such prediction, the context switch prediction unit being adapted to analyze one or more instructions within a context switch inducing instruction cache;

where the interrupt handler signals the fetch and decode units to insert micro-ops corresponding to the interrupt servicing instructions into ~~an~~ the instruction stream sequence within the instruction cache without flushing the instruction cache; and

where the interrupt handler is coupled to the prediction unit instructing the fetch and decode units.

14. (Currently amended) An apparatus, comprising:

an instruction queue mechanism to stage an instruction stream sequence including mainline and interrupt instructions for execution;

a plurality of interrupt inputs corresponding to a plurality of hardware input/output devices;

interrupt priority logic to prioritize the plurality of interrupt inputs;

bandwidth allocation logic to allocate processing resources to service the plurality of interrupts according to their relative priority;

a decoder to generate p-code representative of the instruction stream sequence including mainline and interrupt instructions responsive to the plurality of interrupts and the bandwidth allocation logic;

a p-code insertion mechanism to insert the p-code representative of the interrupt instructions into the p-code representative of the mainline instructions in an instruction stream according to the processing resources allocated to the plurality of interrupt inputs;

a p-code processor coupled to the instruction queue mechanism and to the insertion mechanism to execute the p-code representative of the mainline and interrupt instructions in the instruction stream;

a retirement unit to retire executed p-code representative of the mainline and interrupt instructions back to an instruction cache.

15. (Previously presented) The apparatus of claim 14,

where the instruction queue mechanism includes an instruction cache and an in-order instruction unit to perform branch predictions; and

where the p-code processor includes a dispatch and out-of-order execution units to schedule and execute the p-code representative of the mainline and interrupt instructions in parallel among one or more execution ports.

16. (Canceled)

17. (Previously presented) The apparatus of claim 15 where the in-order instruction unit includes said allocation logic.

18. (Previously presented) The apparatus of claim 17 where the allocation logic is coupled to a computer operating system.

19. (Previously presented) The apparatus of claim 14 where the priority logic is coupled to a computer operating system.

20. (Previously presented) The apparatus of claim 14 where the allocation logic is coupled to a current-usage model.

21. (Previously presented) The apparatus of claim 14 where the priority logic is coupled to a current-usage model.

22. (Currently amended) A machine-readable medium comprising a computer-readable medium containing instructions, that, when executed by a machine cause the machine to perform a method comprising:

detecting an interrupt service request;

storing into an instruction cache interrupt service instructions in response to detecting the interrupt service request;

forming an instruction stream sequence in the instruction cache, the instruction stream sequence including mainline program instructions and the interrupt service instructions resulting in allocating core processor bandwidth between the interrupt servicing and mainline program instructions in accordance with capacity of the core processor;

fetching instructions from the instruction stream sequence within the instruction cache;

~~inserting interrupt servicing instructions responsive to the interrupt service request into mainline program instructions within an instruction queue mechanism;~~

processing instructions from the instruction stream sequence within the instruction ~~queue mechanism~~ cache including the ~~inserted~~ interrupt servicing instructions; and

signaling after detecting an impending natural context switch represented by a recognized instruction sequence within the instruction ~~queue mechanism~~ cache;

where detecting an impending natural context switch is predicated on ~~an~~ the instruction stream sequence within the instruction ~~queue mechanism~~ cache of the machine.

23. (Previously presented) The article of claim 22 comprising:

signaling with a vector representing an interrupt service request upon determining that an interrupt service request of a priority that exceeds a given threshold has occurred; and fetching interrupt service instructions from memory in accordance with the vector representing the determined priority interrupt service request.

24. (Canceled)

25. (Currently amended) A computer system, comprising:

one or more input/output (I/O) devices to generate one or more hardware interrupts;

an interrupt concentrator to rank and queue the interrupts into an ordered interrupt array;

a core processor coupled to the interrupt concentrator, the core processor including an instruction cache, fetch unit, and a decode unit,

the instruction cache to store an instruction stream sequence, the instruction stream sequence including mainline program instructions and interrupt service instructions resulting in allocating core processor bandwidth between interrupt servicing and mainline program instructions in accordance with capacity of the core processor;

the fetch unit to fetch instructions from the instruction stream sequence within the instruction cache and

the decode unit to decode the instruction stream sequence including interrupt service instructions and other program instructions into micro-ops,

the core processor including a dispatch unit to schedule the micro-ops, and an execute unit having one or more execution ports to execute the micro-ops in the one or more execution ports; and

an interrupt-handler responsive to the one or more hardware interrupts, the interrupt-handler instructing the fetch and decode units to insert into an instruction sequence decoded micro-ops representing interrupt servicing instructions for scheduling and execution by the dispatch and execute units, wherein the instruction sequence also comprises decoded micro-ops representing other program instructions.

26. (Currently amended) The computer system of claim 25 which supports plural hardware interrupt inputs, where the interrupt-handler includes the interrupt concentrator

to determine priority among and to schedule the plural hardware interrupts, the interrupt-handler instructing the fetch and decode units to insert plural instances of such decoded micro-ops, each instance representing one or more corresponding sets of the interrupt servicing instructions, into the ~~normal~~ instruction sequence for serial scheduling and execution of the plural instances of such decoded micro-ops by the dispatch and execute units in accordance with the determined priority of said plural hardware interrupts.

27. (Previously presented) The computer system of claim 25 comprising:

a context switch prediction unit coupled with the fetch and decode units to predict a naturally occurring context switch represented by a recognized instruction sequence within the instruction cache and to signal the interrupt-handler upon such prediction;

where the interrupt-handler is coupled to such signaling from the prediction unit instructing the fetch and decode units.

28. (Currently amended) A method, comprising:

detecting an interrupt service request;

storing into an instruction cache micro-ops representative of interrupt servicing instructions responsive to the interrupt service request in such manner that the inserted micro-ops representative of the interrupt servicing instructions intervene micro-ops representative of mainline program instructions to form an instruction stream sequence;

allocating core processor bandwidth between the inserted micro-ops representative of the interrupt servicing instructions and the micro-ops representative of the mainline program instructions in accordance with capacity of a core processor;

fetching instructions from the instruction stream sequence within the instruction cache;

~~inserting micro-ops representative of interrupt servicing instructions responsive to the interrupt service request into an instruction queue mechanism in such manner that the inserted micro-ops representative of the interrupt servicing instructions intervene micro-ops representative of mainline program instructions within the instruction queue thereby allocating core processor bandwidth between the inserted micro-ops representative of the~~

~~interrupt servicing instructions and the micro-ops representative of the mainline program instructions;~~

processing ~~the micro-ops representative of the interrupt servicing inserted into the micro-ops representative of the mainline program instructions within~~ the instruction ~~queue mechanism~~ stream sequence~~, where core processor bandwidth allocation between the micro-ops representative of the mainline program instructions and the inserted micro-ops representative of the interrupt servicing instructions is achieved by concurrent in-line staging of the same within the instruction queue mechanism~~ without first flushing the instruction ~~queue mechanism~~ cache ~~of its contents~~; and

re-processing the instruction stream sequence including micro-ops representative of the interrupt servicing instructions ~~inserted into~~ and the micro-ops representative of the mainline program instructions in the one or more out-of-order execution units by reordering and retiring the executed micro-ops including those micro-ops representing the inserted interrupt servicing instructions to ~~an~~ the instruction cache.